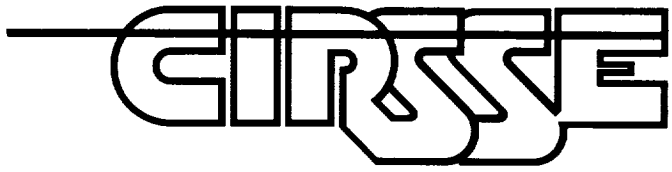

Center for Intelligent Robotic Systems for Space Exploration

Rensselaer Polytechnic Institute
Troy, New York 12180-3590

ANALYSIS OF TERRAIN MAP MATCHING USING MULTISENSING TECHNIQUES FOR APPLICATIONS TO AUTONOMOUS VEHICLE NAVIGATION

by

Lance Page & C.N. Shen

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering
Troy, New York 12180-3590

October, 1990

CIRSSE REPORT #73

Analysis of terrain map matching using multisensing techniques for applications to autonomous vehicle navigation[1]

Lance Page and C. N. Shen

Rensselaer Polytechnic Institute
Department of Electrical, Computer, and Systems Engineering
Troy, New York

ABSTRACT

This paper describes skyline-based terrain matching, a new method for locating the vantage point of laser range-finding measurements on a global map previously prepared by satellite or aerial mapping. Skylines can be extracted from the range-finding measurements and modelled from the global map, and are represented in parametric, cylindrical form with azimuth angle as the independent variable. The three translational parameters of the vantage point are determined with a three-dimensional matching of these two sets of skylines.

1. INTRODUCTION

Semi-autonomous navigation refers to an unmanned vehicle's ability, given its starting position, its destination, and a rough plan to get there, to reach the destination without further instructions. Semi-autonomous navigation typically consists of four main steps, repeated until the destination is reached:

1. Sensing the environment in order to build a three-dimensional *local map*, which contains detailed information about a limited, visible region of the terrain.
2. *Terrain matching*, also known as robot localization[1] or position estimation[2,3], which determines the relationship between the local map and some fixed, global reference. This relationship is expressed as the *vantage*, which is the location and orientation of the vehicle's sensors with respect to some global reference.
3. *Path selection*, which selects the next five to twenty-five meter segment of the journey, based upon mission safety, expediency, fuel conservation, and expected visibility at the next stopping point.
4. Moving the vehicle along the selected path segment.

We have developed a method for performing the terrain matching, suitable for semi-autonomous navigation over a $2\frac{1}{2}$-D unstructured, partially known terrain. "$2\frac{1}{2}$-D unstructured" refers to a terrain on which the ground height is an arbitrary, continuous function of horizontal position. Caves or overhangs violate such a terrain model. "Partially known" means that we are provided a rough, *global map* of the terrain ahead of time. The case of the Mars Rover, on the Mars Sample Return Mission is one such application[4,5,6,7]. In this case, the global map will consist of terrain height samples based on satellite observations made before the Rover's deployment. The samples will likely be on the order of three meters apart[8].

The new technique, called skyline-based terrain matching, uses *skylines* as features which are common to both maps in order to determine the error in the current vantage estimate. "Local skylines" are the skylines, or visual occluding edges, which the vehicle "sees" with its three-dimensional (3-D) sensors. They can be generated simultaneously with the local map, and in fact mark the edges between visible and hidden regions of the terrain. "Global skylines" are the curves that the vehicle expects to see, based on its expected or *a priori* vantage estimate, and the global map. Each iteration of the algorithm generates a set of global skylines for the most recent vantage estimate, compares them to the local skylines, and produces a new vantage estimate.

The vantage has six parameters: three for translation, and three for orientation (yaw, pitch, and roll). In the current version of the terrain matching algorithm, all three orientation parameters are assumed known. For simplicity, we assume that the vehicle's pitch and roll are zero.

[1]Research performed for the NASA Center for Intelligent Robotic Systems for Space Exploration (CIRSSE) at RPI under NASA contract #NAGW-1333.

2. OVERALL TERRAIN MATCHING ALGORITHM

The terrain matching algorithm receives as input:

1. An initial vantage estimate, based on a previous position of the vehicle, and on the path which it has approximately followed from that position.
2. A set of local skylines, extracted from the sensory information.

It also has access to the global map, from which it can generate a set of global skylines for any trial vantage above the terrain.

The algorithm attempts to find x_v, y_v, and z_v, which are the translational parameters of the vantage with respect to a global coordinate system, (x^g, y^g, z^g) fixed to the terrain. As mentioned above, the vantage yaw (θ_v) is assumed known, and pitch and roll are assumed to be zero. θ_v is the direction that the vehicle is facing in the horizontal projection, measured counter-clockwise from the x^g-axis. A local coordinate system, (x^l, y^l, z^l) originates at the vehicle's vantage point, and the relationship between the local and global coordinate systems is a translation:

$$\begin{aligned} x^g &= x_v + x^l \\ y^g &= y_v + y^l \\ z^g &= z_v + z^l \end{aligned} \tag{1}$$

The initial vantage estimate is expressed as $(\hat{x_v}^0, \hat{y_v}^0, \hat{z_v}^0)$, and the vantage estimate after i iterations of the algorithm is $(\hat{x_v}^i, \hat{y_v}^i, \hat{z_v}^i)$. The error in the vantage estimate after i iterations is denoted (x_0^i, z_0^i, z_0^i):

$$\text{For } i \geq 0, \quad \begin{aligned} x_v &= \hat{x_v}^i + x_0^i \\ y_v &= \hat{y_v}^i + y_0^i \\ z_v &= \hat{z_v}^i + z_0^i \end{aligned} \tag{2}$$

The ith iteration of the algorithm consists of the following steps:

1. Generate the global skylines which, according to the rough map, are the skylines visible from the most recent vantage estimate $(\hat{x_v}^{i-1}, \hat{y_v}^{i-1}, \hat{z_v}^{i-1})$.
2. Perform the "curve correspondence," determining which global skylines correspond to which local skylines. Its output is a list of corresponding curve pairs, each pair consisting of one global and one local skyline which represent roughly the same skyline from the two sources of information.
3. Compute the "max-min," which is the longest (max) of the minimum-distance vectors between corresponding pairs of local and global skylines, although the minimum-distance vectors are approximated in the reverse direction by a "normal-to-intersect" procedure.
4. Estimate the vantage error $(\hat{x_0}^{i-1}, \hat{y_0}^{i-1}, \hat{z_0}^{i-1})$, by using the max-min's direction as a rule for point-by-point correspondence and averaging the distances between the curves taken in that direction.
5. Update the vantage estimate by the estimate of its error:

$$\begin{aligned} \hat{x_v}^i &= \hat{x_v}^{i-1} + \hat{x_0}^{i-1} \\ \hat{y_v}^i &= \hat{y_v}^{i-1} + \hat{y_0}^{i-1} \\ \hat{z_v}^i &= \hat{z_v}^{i-1} + \hat{z_0}^{i-1} \end{aligned} \tag{3}$$

The algorithm repeats this cycle until the magnitude of the difference between two successive vantage estimates, that is of the vantage error estimate, is less than a certain threshold. The details of the steps listed above are, except for the curve correspondence, given in the following sections. The curve correspondence algorithm has not yet been implemented and has so far been performed interactively (by a person) rather than automatically.

Before discussing the details of the individual steps, we will define some coordinate systems which will be used to relate data within and between the major steps. From here forward we will drop the i superscripts; by default we will be in the $i+1$th iteration, calculating quantities with the i superscript. A coordinate system called the star ($\star$) coordinate system originates at the vantage estimate, and relates to the global system as follows:

$$\begin{aligned} x^g &= \hat{x_v} + x^\star \\ y^g &= \hat{x_v} + x^\star \\ z^g &= \hat{x_v} + x^\star \end{aligned} \tag{4}$$

The relationships among the global, local, and star coordinate systems are illustrated in Figure 1. It is emphasized that the transformation to the star coordinate system, which is fixed to vantage estimate is known exactly, whereas the transformation to the local coordinate system, fixed to the true vantage, is not. Thus, in order to relate the local skyline data to the global skylines, they are translated into another coordinate system, the dagger (†) system, coincident with the star system. The "dagger curves" are translated copies of the local skylines; the relationship of the dagger curves to the dagger system is identical to that of the local curves to the local coordinates. No computational conversion from local to dagger skylines actually takes place; this "translation" of the local skylines is merely a distinction in the geometric interpretation of how the data is processed.

In spite of differences enumerated below, the local (l) and global ($\star$) curves are taken as approximations of the same features from different sources of information, and thus the local and global curves should be roughly identical. To compare the skylines computationally, they are "overlaid" together, in the star and dagger coordinate systems. If the vantage estimate has no error, then the translation of the local skylines to dagger coordinates is null, and the star and dagger curves should remain approximately coincident; if an error is present in the vantage, then the star and dagger curves should differ by a translation equal to the error.

There are three causes of differences between the local and global skylines, even before translating the local curves into dagger curves:

1. Errors in the global map cause the actual (local) and predicted (star) skylines to differ.
2. Any error in the vantage estimate causes differences between the curves, because skylines are vantage-dependent.
3. Numerical errors from sampling and interpolation, both in the generation of global skylines and in the subsequent spline interpolation of both sets of skylines.

As long as the vantage estimate improves in each iteration of the algorithm, the differences resulting from the second cause listed will diminish.

Throughout this report, each cylindrical coordinate system is related to its corresponding rectangular coordinate system (the one with the same superscript) as follows: θ is measured counter-clockwise from the x-axis; ρ is the length of the horizontal projection; and z is the same as the rectangular z. Mathematically,

$$\begin{aligned} \theta &= \text{atan2}(x, y) \\ \rho &= \sqrt{x^2 + y^2} \\ z &= z \end{aligned} \tag{5}$$

where the $\text{atan2}(x, y)$ function is a special version of $\arctan(y/x)$, which insures that the resulting angle is in the appropriate quadrant.

3. GENERATING SKYLINES

In the most intuitive terms, the skyline curves are the basic curves which we would draw in a crude depiction of a scene, especially mountains or "rolling hills." Skylines are the occluding edges, or local horizons, of the terrain that are visible from some vantage point above the terrain. They are 3-D curves, even though they are characterized by their behavior on the 2-D image projections onto their vantages.

Geometrically, skylines on $2\frac{1}{2}$-D terrain can be defined as the connected sets of points on the terrain where, in terms of spherical coordinates about the vantage point, the following conditions hold: first, that there is a local maximum of elevation angle with respect to radial distance from the vantage; and second, that the point is not obscured by a skyline curve crossing the same azimuth, which is closer to the vantage and has a higher elevation angle.

Skyline curves are approximated from a rough map of a terrain according the geometric definition above, for comparison to the skyline curves observed by the robot's sensors. Since the set of skylines obtained depends upon the vantage they are observed from, the rough map skylines are generated for the current "best estimate" of the vantage point, $(\hat{x_v}, \hat{y_v}, \hat{z_v})$. Normally, the vehicle's vantage point is assumed to be two meters above the terrain.

The algorithm for collecting skyline points iterates through a series of azimuths which span a predefined range of directions centered at the vehicle's yaw, and which are separated by predefined intervals. Each azimuth is processed by the following steps:

1. "Look up" a series of terrain height samples $z_m(x, y)$ from the rough map, along a line which in the horizontal projection starts at the vantage and proceeds in the direction of the azimuth. This generates a planar cross-section, or *terrain section* of the environment according to the map.

2. Along that section, detect local peaks of *elevation angle* β with respect to horizontal distance ρ. (It can be shown that a maximum in β with respect to ρ is equivalent to a maximum in β with respect to radial distance r.) A local peak is detected when one sample elevation angle is greater than both of its neighboring samples. A quadratic function of β vs. ρ is fitted to each of the three-sample sequences, and the maximum of each quadratic function is calculated and stored as a ridge point.

3. The ridge points are converted into cylindrical, star coordinates as follows:

$$\begin{aligned} \theta^\star &= \text{(azimuth for this terrain section)} \\ \rho^\star &= \rho \\ z^\star &= \rho \tan\beta \end{aligned} \tag{6}$$

   where ρ and β are the horizontal distance and elevation angle, respectively, from the vantage to each skyline point.

Figure 2 shows a terrain section in which two ridge points have been found.

As collected above, the skyline points comprise a big set of individual points. The next step is to group the points into curves, where each curve is represented by an ordered set of points. Points are "connected" into curves according to two criteria:

1. The two points are from adjacent terrain sections in terms of how the ridge points were collected, but not from the same one.

2. The difference between the ρ's of the two points must not exceed a certain portion of their average ρ.

The implementation takes advantage of the fact that the ridge points are collected monotonically in azimuth.

After both sets of skylines are obtained, the curve correspondence must be performed, to associate curves from one set with specific curves in the other. As will be shown in the Results section, multiple correspondences where more than one local skyline corresponds to the same global skyline or vice-versa are possible. In the following sections, references to the sets of skylines are implicitly limited to those skylines which were found to have corresponding curves in the other set.

Following curve correspondence, each skyline is interpolated into a smooth curve using natural cubic splines of ρ and z parameterized by θ. The splines result in C2-continous functions[2] $\rho(\theta)$ and $z(\theta)$ passing through the individual skyline points collected from either the global map or the sensor data.

4. MAX-MIN

The max-min is an approximation of the longest vector which is the shortest vector from a point on one skyline to its corresponding skyline. It has been found to usually provide a rough but useful approximation to the vantage error. To filter the noise that is present, however, only the max-min's direction is used. The estimated length of the translation is an average distance between the curves, measured specifically in the max-min's direction. The averaging procedure will be described in the next section.

The reason that the max-min approximates the translation is explored in [9]. With ideal, identical curves, differing only by a translation, the max-min can not be longer than the translation between them. As assumptions about the curves are relaxed, permitting non-corresponding endpoints and noisy data, the principle no longer strictly holds, but we still find that the max-min is usually a useful estimate. Exceptions to this are possible under certain circumstances, with examples presented in [9].

Shortest-distance approximations are taken from evenly spaced points on all the curves, to their corresponding curves. The approximations are calculated both from global skylines to local skylines and vice-versa. Because computing the shortest-distance vector from a point to a 3-D cylindrical cubic spline is fairly costly, two simplifications are made. First, only the horizontal projection of the curves is used in computing the max-min, thereby reducing the computations to two dimensions. The second simplification stems from the observation that, except when the shortest-distance vector leads to an end-point of the corresponding curve (a case which would be discarded anyway), the shortest-distance vector is normal to its terminal, or arrow-head point. A vector which is normal to a curve at its *originating point*, and ends at an intersection with the corresponding curve can approximate a shortest-distance vector, in the *reverse direction*. Such vectors are called *normal-to-intersect* vectors. Computing a normal-to-intersect vector requires little more than a search along a line for an intersection with the other curve, as will be shown.

We now describe the normal-to-intersect procedure, for intersecting the normal from a point on a dagger (local) curve, to the corresponding star (global) curve. Computing a normal-to-intersect in the reverse direction is identical except for swapping the curves.

Define ϕ as the angle that the dagger curve at $(\theta_n, \rho^\dagger(\theta_n))$ makes with a curve of constant ρ through that point.

$$\phi = \arctan\left(\frac{d\rho^\dagger(\theta_n)}{\rho^\dagger(\theta_n)d\theta}\right) \tag{7}$$

(ϕ is always between -90° and 90°.) The line tangent to the curve points in the direction of $(\theta_n - \phi + 90°)$, and the normal line points in the direction of $(\theta_n - \phi)$. Next, parameterize the normal line in terms of ϵ, which is the distance along the line in the normal direction. Letting the l subscript denote "line," the normal line is parameterized in cylindrical coordinates about the dagger origin as:

$$\rho_l(\epsilon) = \sqrt{(x_n + c_x\epsilon)^2 + (y_n + c_y\epsilon)^2} \tag{8}$$

[2] C2-continuous means that not only function $\rho(\theta)$ is continuous, but also the first and second derivatives of ρ with respect to θ are continous. The same is true for $z(\theta)$ and its first and second derivatives.

$$\theta_l(\epsilon) = \text{atan2}((x_n + c_x\epsilon), (y_n + c_y\epsilon)) \tag{9}$$

where

$$\begin{aligned} x_n &= \rho^{\dagger}(\theta_n)\cos\theta_n \\ y_n &= \rho^{\dagger}(\theta_n)\sin\theta_n \end{aligned} \tag{10}$$

and

$$\begin{aligned} c_x &= \cos(\theta_n - \phi) \\ c_y &= \sin(\theta_n - \phi) \end{aligned} \tag{11}$$

At the point where the normal line intersects the star curve, $\rho_l(\epsilon) = \rho^{\star}(\theta_l(\epsilon))$. Thus, define

$$f(\epsilon) = \rho^{\star}(\theta_l(\epsilon)) - \rho_l(\epsilon) \tag{12}$$

and our task is to search for $\epsilon : f(\epsilon) = 0$. This search is performed numerically using the `zbrac()` and `zbrent()` subroutines of [10]. In some cases, the normal lines do not intersect the corresponding curves at all; when this occurs, the normal-to-intersects do not exist and are not used in any subsequent calculations.

Normal-to-intersect calculations are made from evenly spaced points on all curves, to their corresponding curves. The normal-to-intersect vectors are calculated in both directions— from dagger to star and vice-versa— because the normal-to-intersect distances sometimes exhibit distinct peaks only in one of these directions.

The normal-to-intersect vector with the overall biggest length (magnitude of ϵ) is selected as the max-min; it is the longest normal-to-intersect vector. The max-min is represented by its ϵ, and its c_x and c_y of Equ. 11 which represent the direction of the normal-to-intersect vector. The direction in terms of whether it is a dagger-to-star normal-to-intersect or a star-to-dagger curve normal-to-intersect is also noted.

5. TRANSLATION

This section describes how the vantage error (x_0, y_0, z_0) is estimated, once the max-min is computed. To repeat, the vantage error is assumed to be a translation between from the dagger to the star skylines.

The direction of the max-min is effectively used as a rule of point-by-point correspondence between the two sets of curves. That is, we retain the direction of the max-min but not its length, and the length of the translation is then estimated as the *average* of the distance between the curves, taken in this direction. A distance from a point on one curve to its corresponding curve, taken in a particular direction, is called a *direction-to-intersect* measurement.

The equations for the horizontal translation are:

$$\hat{x_0} = c_x \frac{1}{\left(N_{\dagger\star} + N_{\star\dagger}\right)} \left(\sum_{i=1}^{N_{\dagger\star}} \Delta_{\dagger\star i} - \sum_{i=1}^{N_{\star\dagger}} \Delta_{\star\dagger i} \right) \tag{13}$$

$$\hat{y_0} = c_y \frac{1}{\left(N_{\dagger\star} + N_{\star\dagger}\right)} \left(\sum_{i=1}^{N_{\dagger\star}} \Delta_{\dagger\star i} - \sum_{i=1}^{N_{\star\dagger}} \Delta_{\star\dagger i} \right) \tag{14}$$

where each Δ_i is a direction-to-intersect measurement from one curve to another; $N_{\dagger\star}$ and $N_{\star\dagger}$ are the number of direction-to-intersects which intersected in the respective directions. The star-to-dagger measurements are negated for purposes of calculating $\hat{x_0}$ and $\hat{y_0}$. Computationally, the direction-to-intersects are equivalent to normal-to-intersects, but their directions represented by c_x and c_y are pre-specified from the max-min rather than calculated with Equ. 11 for each measurement.

Iter.	$\hat{x_v}$	$\hat{y_v}$	$\hat{z_v}$	Remaining Error Magn.	Incremental Improvement	Overall Improvement
0	0.00	40.00	7.00	2.89		
1	1.43	38.35	6.27	0.69	76.1%	76.1%
2	1.48	38.12	6.38	0.54	22.4%	81.3%
3	2.13	37.82	6.51	0.23	56.5%	92.0%
4	2.10	37.95	6.44	0.11	52.4%	96.2%
5	2.09	37.96	6.43	0.10	11.8%	96.5%

Table 1: Progress in each iteration of Example 1. Actual vantage (2, 38, 6.43).

Example 1 was the following problem: the vehicle's true vantage was $(x_v, y_v, z_v) = (2, 38, 6.43)$, with a yaw of 90°, or facing north. A graphical depiction of the vehicle at the true vantage is shown in Figure 3, and this vantage is two meters to the right, and two meters forward of the top of the hill in this view. (Vehicle is drawn with the actual pitch and roll for its position, but pitch and roll are neglected in the analysis. The vantage is assumed to be two meters above the terrain, regardless of the gradient of the terrain on which the vehicle lies.) The terrain height at $x^g = 2$, $y^g = 38$ is $z^g = 4.43$, and the vantage is two meters above the terrain. The vehicle was given an initial vantage estimate at precisely two meters above the top of the hill, $(\hat{x_v}^0, \hat{y_v}^0, \hat{z_v}^0) = (0, 40, 7.00)$. Before calling the terrain matching algorithm, the program generated skylines from the actual terrain function for the actual vantage, to serve as the local skylines. The algorithm was given the local skylines in cylindrical coordinates about the local vantage, and the estimated vantage.

This example took five iterations before termination, and the effects of each iteration are recorded in Table 1. In the table, the error magnitude is the distance between the true vantage and the vantage estimate. Percentage improvements were calculated as [1 - (new error magn.)/(old error magn.)](100%). Figure 4 graphs the horizontal projection of the sequence of vantage estimates, as the algorithm grew closer to the true vantage. The dotted lines show the range of azimuth in which skylines were collected, which was ±60° from the yaw. Each 'x' represents one of the vantage estimates in Table 1. From its initial estimate of (0, 40, 7.00), the algorithm settled at an estimate of (2.09, 37.96, 6.43).

Figures 5 and 6 show the corresponded curves at the initial and final vantage estimates, illustrating the curve "matching" accomplished by the algorithm. Figures 5.a and 6.a show the horizontal projections of the curves, and Figures 5.b and 6.b show height ($z^\star = z^\dagger$) vs. azimuth ($\theta^\star = \theta^\dagger$). The solid lines indicate global skylines, and the dotted lines indicate local skylines. Only the corresponded curves are shown. Note the multiple correspondence in the initial sets, in which two global skylines corresponded to the same local skyline. Also note that a local skyline (at a horizontal ρ of about five meters) which did not have a corresponding global skyline at the initial estimate, did have one at the final estimate. The two sets of skylines grew more similar as the vantage estimate grew more accurate.

The correspondence shown in Figure 6 was performed for illustration only and was not part of the algorithm's normal execution. The algorithm terminated immediately after determining that the fifth iteration made a sufficiently small vantage update, and did not process the global skylines at its final vantage estimate. That is, these would be the corresponded curves of a hypothetical sixth iteration.

More figures in the form of Figures 4 and 5 are shown for Examples 2–4 in Figures 7–9, respectively. The algorithm's performance on the these examples is summarized in Table 2. In Example 2, the vehicle faced a yaw of 0° (west), lying one meter north and one meter west of the origin (the verticle line coming out of the terrain in Figure 3; the initial estimate was one meter south and one meter east of the origin. In this example, the second iteration was particularly successful, even though the first iteration made only marginal progress (19.1%). The first iteration brought the vantage estimate close enough for an excellent match to exist in the second iteration.

Example	Local (True) Vantage	Init. Vantage Estimate	Yaw	Init. Error	First Iter. Improvement	Iters. to Termination	Final Error
1	(2., 38., 6.43)	(0., 40.0, 7.00)	90°	2.89	76.1%	5	0.10
2	(1., 1., 6.85)	(-1., -1., 6.85)	0°	2.83	19.3%	3	0.03
3	(0., 11., 1.43)	(0., 9., 2.57)	45°	2.30	70.3%	4	0.09
4	(0., -4., 5.86)	(0., -6., 4.63)	90°	2.35	70.5%	5	0.04

Table 2: Performance summary for Examples 1–4.

Example 3 put the vehicle about ten meters north of the origin, at the steepest part of the descent in the valley, facing northwest. The bulk of the progress in this example was made in the first two iterations; the vantage estimates continued to improve at a consistent but less dramatic rate after that.

In Example 4, the vehicle was facing north, ascending the hill at the origin from the south. Once again, there were some iterations— in this case, particularly the second— which produced only marginal progress, but this progress was sufficient to facilitate better matching in successive iterations.

7. CONCLUSION

We have described an algorithm for skyline-based terrain matching, based on a "max-min principle" (which is elaborated further in [9]). In each iteration, an error in the vantage estimate is assumed to manifest itself as a translation between the star curves (from the global map) and the dagger curves (from the sensors), when the respective (estimated and true) vantages are translated onto each other. The horizontal direction of this translation is approximated as the direction of the max-min, and the horizontal length of this translation is approximated by averaging the distances between the curves in this horizontal direction. The vertical component of the translation is also found by using the max-min's direction as a rule of point-by-point correspondence, and averaging the differences between corresponding curves. Each iteration of the new algorithm generates a new estimate for the vehicle's vantage point, by adding the estimated 3-D translation between the skylines to the previous estimate. In successive iterations, the local and global skylines grow more similar as the vantage estimate grows more accurate.

The algorithm has performed well on a variety of test cases, such as the examples given in Section 6. These examples demonstrate that even iterations which provide marginal improvement are valuable, because the resulting skylines at the new vantage estimate match the local skylines even better.

A limitation of skyline-based terrain matching is the need for enough skylines to be visible from a vantage to contain the required 3-D information. This depends on two factors: first, the azimuth range of skyline collection by the sensors and the rough map ($\pm 60°$ in our examples) must be wide enough. A good max-min depends upon at least one corresponded skyline to produce a normal-to-intersect measurement in the direction of the translation between the curves. Increasing the azimuth range of skyline collection increases the chances of some skylines having normals in the direction of the actual translational error. Second, it is up to the particular terrain on which the vehicle lies to provide adequate skylines for the algorithm to collect and use. A terrain which is simply too flat and non-descript will provide no useful skylines for reference. We can control the first factor, but the second depends solely on the environment in which the algorithm is applied.

In its current formulation, the algorithm is somewhat susceptible to errors in the max-min by simply taking the single biggest normal-to-intersect. As the azimuth width of skyline collection is increased beyond about $\pm 75°$, certain anomalies (described in [9]) become more common because of long, twisty skylines with multiple correspondences which arise. Such skylines contain a great amount of information, but also a lot of ways in which strange relationships may appear. In future experiments, additional conditions on the max-min may be necessary in order to handle these occasional anomalies.

REFERENCES

1. Eric Krotkov. Mobile robot localization using a single image. In *Proc. IEEE Robotics and Automation Conference*, pages 978–983, 1989.

2. C. Ming Wang. Location estimation and uncertainty analysis for mobile robots. In *Proc. IEEE Robotics and Automation Conference*, pages 1230–1235, 1988.

3. Paul R. Klarer. Autonomous land navigation in a structured environment. *IEEE Aerospace and Electronic Systems Magazine*, 5(3):9–11, March 1990.

4. C. N. Shen and Lance Page. Fusion of gross satellite sensing and laser measurements by skyline map matching for autonomous unmanned vehicle navigation. In *Proc. SPIE Optical Engineering and Aerospace Sensing Symposium*, Orlando, Florida, April 1990.

5. C. N. Shen and George Nagy. Autonomous navigation to provide long-distance surface traverses for mars rover sample return mission. In *Proc. Fourth IEEE International Symposium on Intelligent Control*, pages 362–367, Albany, NY, September 1989.

6. Brian Wilcox and Donald Gennery. A mars rover for the 1990's. *Journal of the British Interplanetary Society*, 40:484–488, 1987.

7. Donald Gennery. Visual terrain matching for a mars rover. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, California, June 1989.

8. J. C. Mankins (ed.). Mars rover technology workshop proceedings. Technical report, Jet Propulsion Laboratory, April 1987. JPL D-4788.

9. C. N. Shen and Lance Page. Skyline-based terrain matching using a max-min principle. Technical report, NASA Center for Intelligent Robotic Systems for Space Exploration, Rensselaer Polytechnic Institute, October 1990. CIRSSE Report #72.

10. William H. Press, Brian R. Flannery, Saul A. Taukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

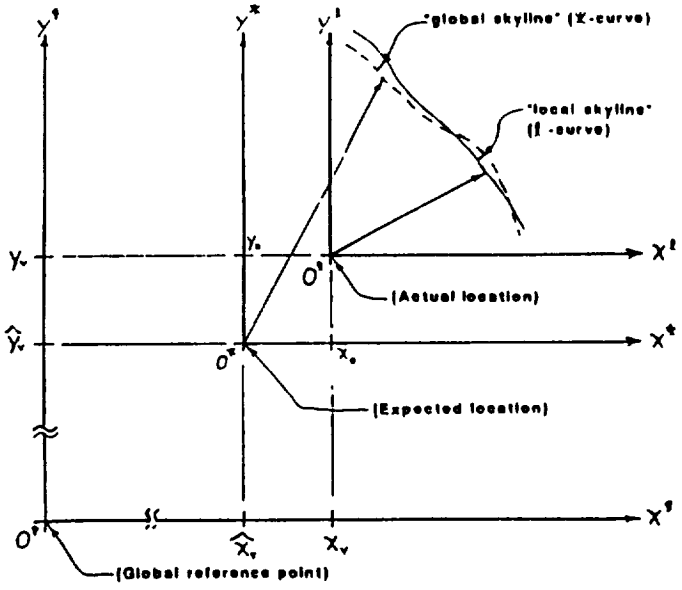

Figure 1: Relationships among the global(g), local (l), and star (s) coordinate systems.

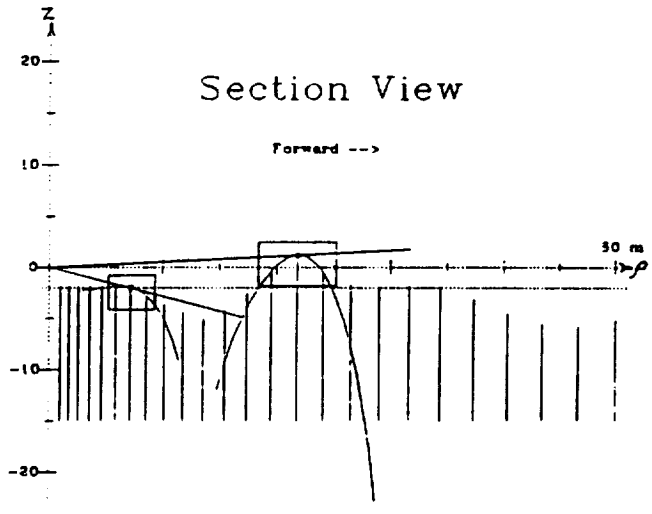

Figure 2: Example terrain section, showing two skyline points.

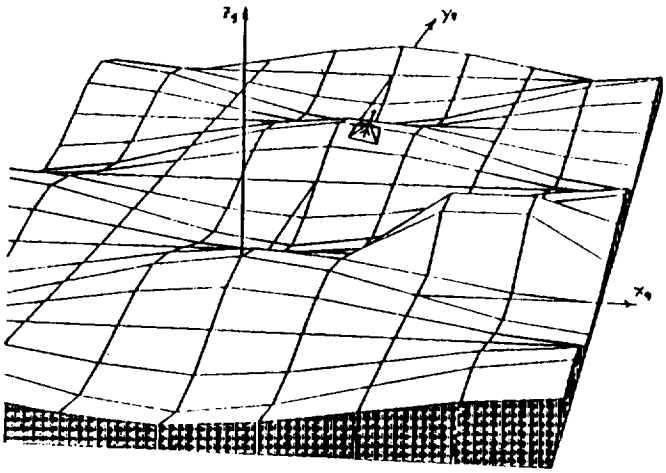

Figure 3: Example 1: the "true" location of the vehicle.

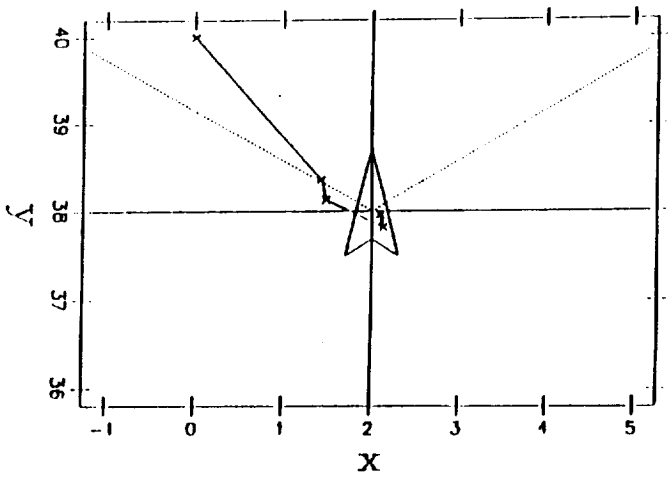

Figure 4: Example 1: sequence of vantage estimates.

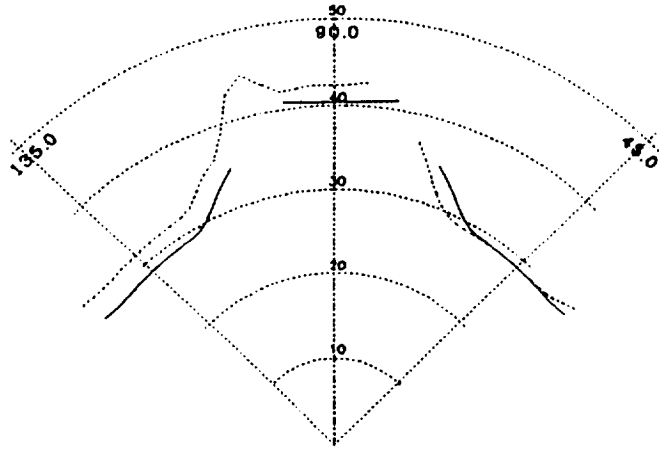

Fig. 5a: Horizontal projections.

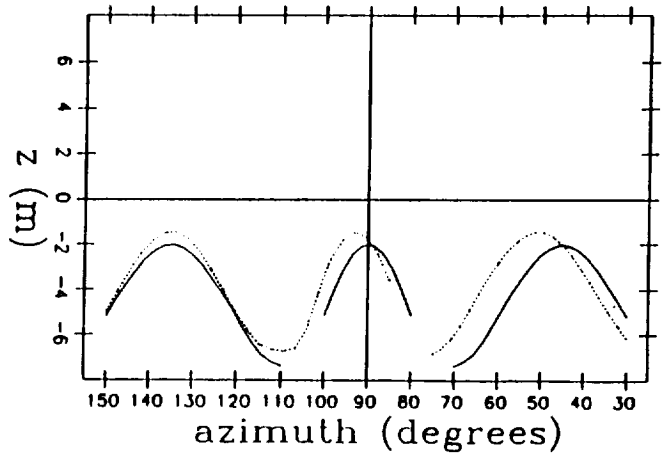

Fig. 5b: $(z^* = z^\dagger)$ vs. $(\theta^* = \theta^\dagger)$.

Figure 5: Example 1: corresponding curves for the initial estimate.

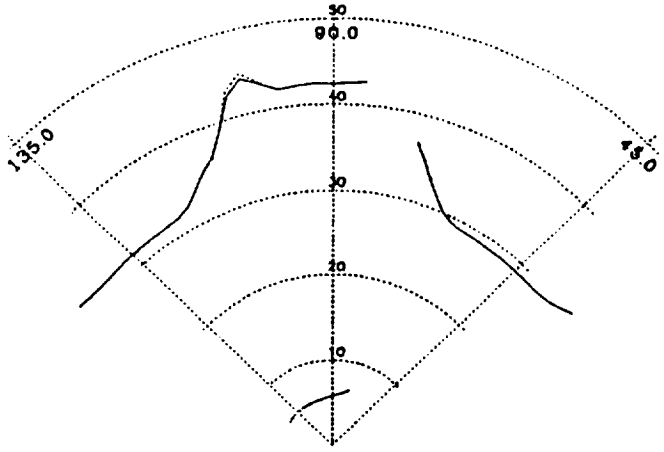

Fig. 6a: Horizontal projections.

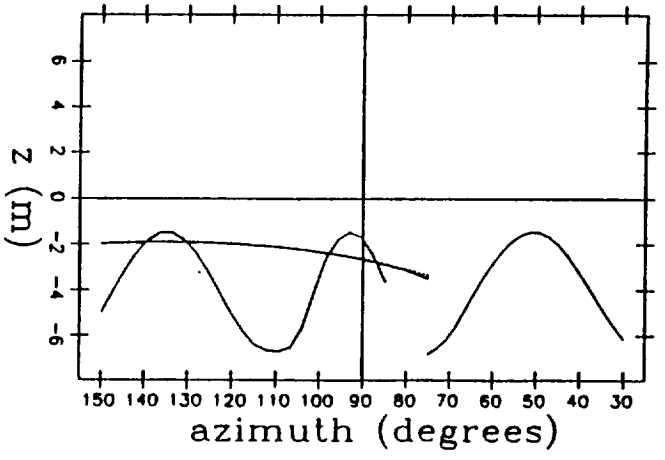

Fig. 6b: $(z^* = z^\dagger)$ vs. $(\theta^* = \theta^\dagger)$.

Figure 6: Example 1: corresponding curves for the final estimate.

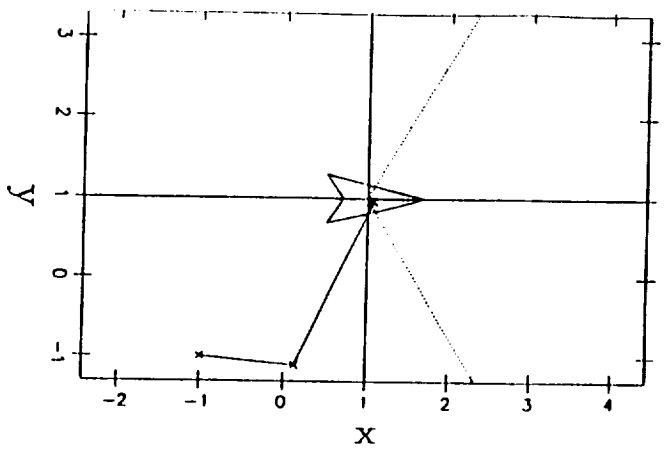

Fig. 7a: Algorithm performance for Example 2.

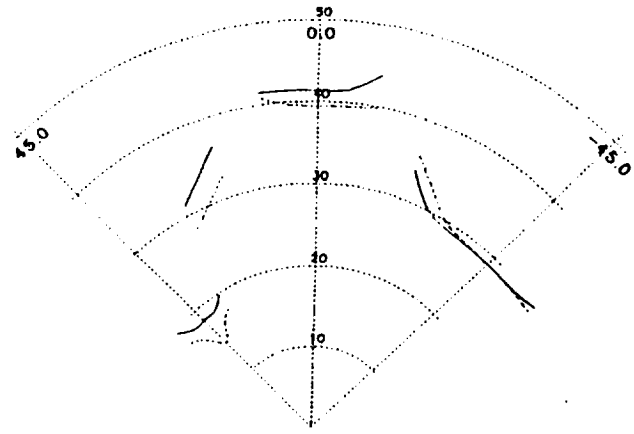

Fig. 7b: Corresponded curves in the first iteration.

Figure 7: Example 2.

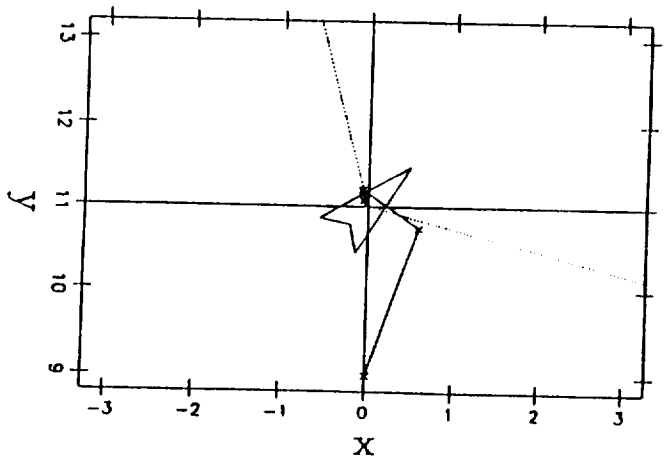

Fig. 8a: Algorithm performance for Example 3.

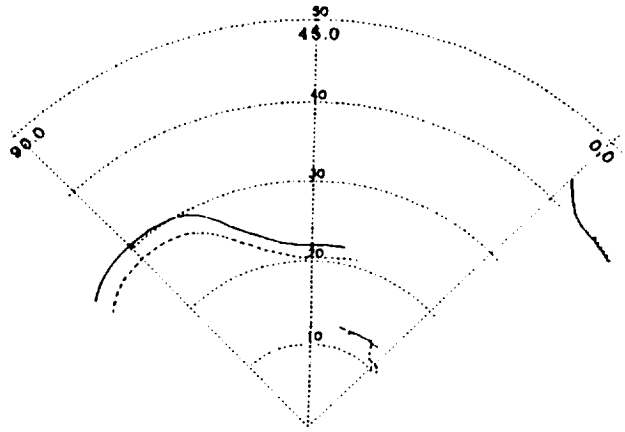

Fig. 8b: Corresponded curves in the first iteration.

Figure 8: Example 3.

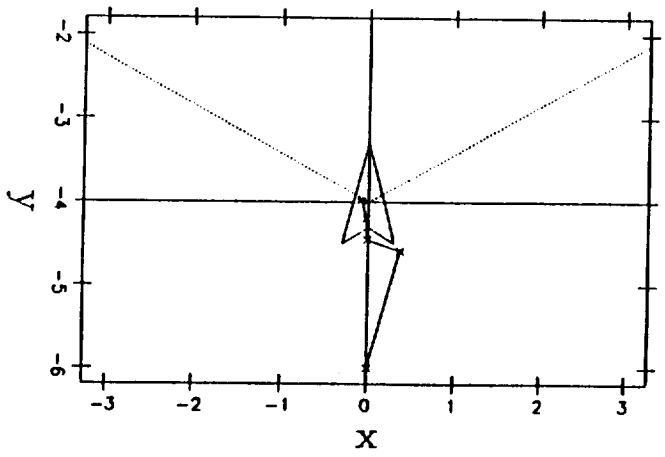

Fig. 9a: Algorithm performance for Example 4.

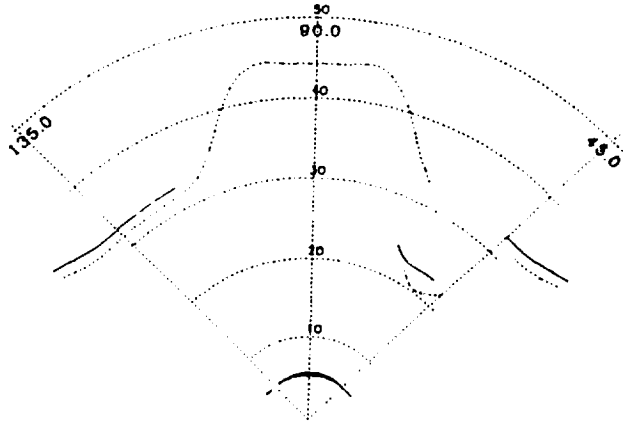

Fig. 9b: Corresponded curves in the first iteration.

Figure 9: Example 4.